ECE/CS 584: Embedded and cyberphysical system verification	Fall 2025
Your name, netid	*Homework 3: Neural Verification and Dynamical systems*
	Due October 12th 11:59pm

Upload your writeup (.pdf) and code (.py) to Gradescope. In the writeup, put each answer on a separate page and label it with the correct number. Ensure your code runs on the python environment described in Problem 3. You may not get points if your code does not run. Code for this homework can be downloaded here.:
https://uofi.box.com/s/ugwhbjik45exxbn10ekz9n8g12pwllel

Problem 1. (25 points). Given system matrix $A = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}$, we consider an 2-dimensional discrete-time linear automaton

automaton $LinearSys()$
variables :
$x : \mathbb{R}^2$
actions :
$step()$
transitions :
$step()$
$x := Ax$

The set of initial states $\Theta \subset \mathbb{R}^2$ is a convex combination of four vertices $v_0^{(1)} = (1,1), v_0^{(2)} = (1,2), v_0^{(3)} = (2,2), v_0^{(4)} = (2,1)$.

(a) Compute the set of reachable states at time $k = 3$, denoted as $Reach(\Theta, [3,3])$. [Hint: From Problem 3 of Hw2, recall that for a linear system, if its initial set is a convex set, then its reachable set after any number of step must be a convex set.]

(b) Given an unsafe region: $U = \{(x,y) \in \mathbb{R}^2 | y \geq x\}$, we call this automaton is safe at time $k \in \mathbb{N}$, if $Reach(\Theta, [k,k])$ has no intersection with U. Formulate the task of checking whether $Reach(\Theta, [3,3])$ is safe as a SMT problem; that is, express the existence of a point in $Reach(\Theta, [k,k])$ that lies in U as a set of constraints for SMT solver.

(c) An alternative approach is to measure the signed distance from the reachable set to the boundary of the unsafe region $\partial U = \{(x,y) \in \mathbb{R}^2 | x = y\}$. Formulate the safety check as a linear programming problem that minimums the distance from any point in $Reach(\Theta, [3,3])$ to ∂U. If the optimal distance is positive, then the reachable set is disjoint from unsafe region. [Hint: Given a point $(x_0, y_0) \in \mathbb{R}^2$, its absolute distance to the line $ax + by + c = 0$ can be computed by $\frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$]

(d) Please plot the unsafe region boundary ∂U and $Reach(\Theta, [0,5])$ on 2D-plane.

Problem 2. (25 points). In this problem, you are required to use DPLL procedure to solve a SMT problem: $(-x \leq -1) \wedge (y \leq 3) \wedge ((\frac{1}{2}x + y \leq 1) \vee (-2x + y \leq -1)) \wedge (x - y \leq 0)$. A boolen expression of this SMT problem can be written as $a \wedge b \wedge (c \vee e) \wedge d$, where

$$
\begin{aligned}
&a : -x \leq -1; \\
&b : y \leq 3; \\
&c : \frac{1}{2}x + y \leq 1; \\
&d : x - y \leq 0; \\
&e : -2x + y \leq -1
\end{aligned}
$$

(a) First We provide the clauses $\{a, b, c \vee e, d\}$ to DPLL, and it returns model $\{a, b, c, d\}$. Now please concretize it and check it is UNSAT via simplex algorithm. When performing the simplex procedure, pivot the non-basic and basic variables in the following order: $x, y, s_1, s_2, s_3, s_4, s_5$. The below gives you the first step of simplex algorithm.

$$
\begin{aligned}
&s_1 = -x \\
&s_2 = y \\
&s_3 = \frac{1}{2}x + y \\
&s_4 = x - y \\
&s_1 \leq -1 \\
&s_2 \leq 3 \\
&s_3 \leq 1 \\
&s_4 \leq 0 \\
&-\infty < x, y < \infty \\
&\text{Initialize} \langle x = 0, y = 0, s_1 = 0, s_2 = 0, s_3 = 0, s_4 = 0 \rangle \\
&\text{Constraint on } s_1 \text{ is not satisfied} \\
&\text{Pivot non-basic variables } \{x, y\} \text{ to } \{y, s_1\} \\
&\text{Update x value: } 0 + \frac{-1 - 0}{-1} = 1
\end{aligned}
$$

[Hint: The algorithm reports UNSAT only when it encounters a repeated pair of non-basic variables.]

(b) Then we send the clauses $\{a, b, c \vee e, d, \bar{a} \vee \bar{b} \vee \bar{c} \vee \bar{d}\}$ to DPLL, and it returns model $\{a, b, \bar{c}, d, e\}$. Now please concretize it and check it is SAT via simplex algorithm.

(c) Please plot the boundaries of the constraints $\{a, b, c, d, e\}$ and shade the SAT region of ϕ on 2D-plane and highlight the feasible point (x, y) you obtain from part (b).

Problem 3. (35 points) In this problem, you will implement linear programming (LP) formulation and mixed integer linear programming (MILP) formulation on a simple neural network. Please download files `mip.py`, `lp.py`, `data1.pth`, `data2.pth`, `model.pth` and put them under the same folder. **Please make sure your code can run in the python version:3.10**.

You will use the Gurobi optimizer to solve MILP and LP problems. Your code needs to formulate the verification problem using the Python interface of Gurobi. You can refer to the Gurobi Python documentation available at this link (a simple installation command is `pip install gurobipy`).

The code loads a simple pretrained model with three linear layers, and one test image (a handwritten digit, 0–9) from the MNIST dataset. The test image $t \in \mathbb{R}^{784}$ contains 28×28 pixel values. Element-wise perturbation with a given perturbation size is added to each pixel: e.g., when the perturbation size is s, the input set is $S := \{x \mid t_i - s \leq x_i \leq t_i + s,\ \forall i \in [1, 784]\}$. The neural network predicts scores $y \in \mathbb{R}^{10}$, and y_i represents the score for the image being recognized as a digit i. If the label of test image t is c, we want to verify that y_c is always the top-1 score for all $x \in S$ by solving 9 optimization objectives.

You can run code template `mip.py` with an input data `data1.pth` by running `python3 mip.py data1.pth`. (Similiarly, `python3 lp.py data1.pth`) The implementation consists of the following components:

- `class SimpleNN`: Defines a three-layer ReLU neural network structure. **No changes are required.**
- `class Verify`: Provides the procedures for solving the MIP or LP problems.
  - `def create_input`: Defines the range of input variables. Fill in the `TODO` lines using `self.gurobi_model.addVar`.
  - `def add_linear_layer`: Adds constraints between the output and input of a linear layer. Fill in the `TODO` lines using `self.gurobi_model.addVar`.
  - `def add_relu_layer`: Adds constraints capturing the relationship between the input and output of a ReLU operation. Fill in the `TODO` lines and consider three cases: *active* (input always positive), *inactive* (input always non-positive), and *unstable* (all other situations). To add new constraints, use `self.gurobi_model.addConstr`.
  - `def solve_objectives` and `def get_verification_objectives`: Generate concrete objectives and verify them via the Gurobi model. **No changes are required.**
- `def load_model_and_data` and `def extract_model_weight`: Load the input data and assign the weights to the target neural network. **No changes are required.**
- `def verify`: Defines the complete verification procedure and output log information. **No changes are required.**

Now, complete the following tasks:

(1) Consider a ReLU activation function $\hat{z} = ReLU(z)$, for which the pre-activation satisfies $l \leq z \leq u$. Please derive constraints for three cases: *active* ($l \geq 0$), *inactive* ($u \leq 0$), and *unstable* ($l < 0 < u$). For the unstable case, you will need a selector p (MILP: $p \in \{0, 1\}$, LP relaxation: $p \in [0, 1]$, see slide 28&29 of Lecture 8). These results will be applied to fill in TODO lines in `def add_relu_layer`.

(2) Complete the code for the MILP and LP formulation using the given template. The parts that you need to finish have been marked with TODOs in source files (These TODO lines are same for both `mip.py` and `lp.py`). Submit your completed program `mip.py`.

(3) Discuss the performance differences between the MIP and LP approaches for solving this problem, considering factors such as solution accuracy, computational time, and scalability.

(4) Using the data `data1.pth`, collect the exact solution found by the MILP formulation and the lower bound found by the LP formulation for perturbation size $s \in \{0.001, 0.003, 0.01, 0.03, 0.1\}$. Plot the results showing the gap between the exact solution and the lower bound at different perturbation sizes for all 9 optimization objectives. If program does not finish within 10 minutes, you can report "timeout".

Problem 4. (15 points). Consider the autonomous dynamical system $\dot{x} = f(x, t)$, with initial set $\Theta \subseteq val(X)$. Suppose that $V : val(X) \to \mathbb{R}$ is a Lyapunov function for the system. Show that for any sublevel set S of V, if $\Theta \subseteq S$, then S is an inductive invariant of the system.